Advanced Java T.T.

Parallel Streams

Develop Intelligence

Simon Roberts

Join Us in Making Learning Technology Easier

Develop Intelligence

Our mission...

Over 16 years ago, we embarked on a journey to improve the world by making learning technology easy and accessible to everyone.

...impacts everyone daily.

And it's working. Today, we're known for delivering customized tech learning programs that drive innovation and transform organizations.

In fact, when you talk on the phone, watch a movie, connect with friends on social media, drive a car, fly on a plane, shop online, and order a latte with your mobile app, you are experiencing the impact of our solutions.

Upskilling and Reskilling Offerings

Develop Intelligence

Intimately customized learning experiences just for your teams.

- **Workshop** — 2-3 day upskilling experiences
- **Fast Track** — 5-day reskilling experiences
- **Learning Spike** — 1-day technology overviews
- **Target Topics** — 90-minute instructor-led micro-learnings
- **Hack-a-thon** — Learn and build an MVP in 2-3 days

BACK END DEVELOPMENT

BIG DATA

CLOUD COMPUTING

DEVOPS

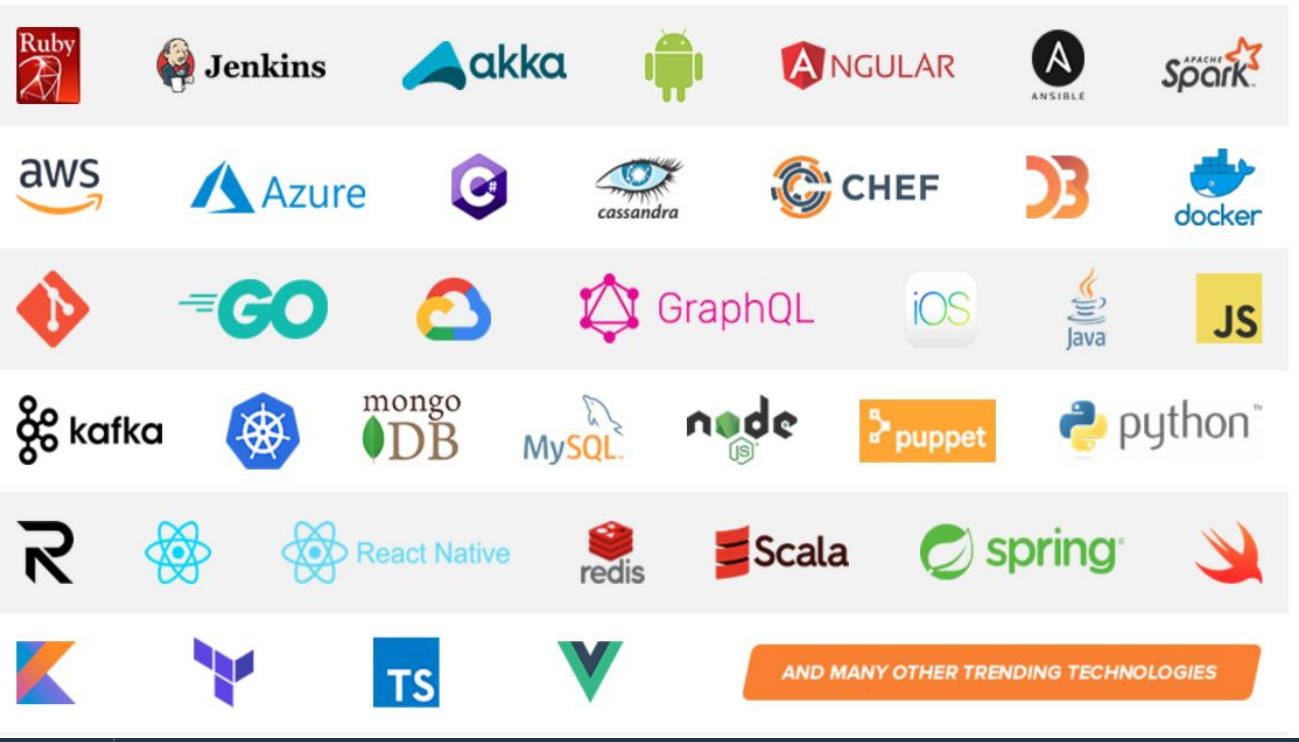
FRONT END DEVELOPMENT

MACHINE LEARNING

MOBILE APP DEVELOPMENT

SOFTWARE ENGINEERING

SYSTEM ADMINISTRATION



World Class Practitioners

Recording Policy

Recordings are provided to participants who have attended the training, in its entirety. Recordings are provided as a value-add to your training, and should not be utilized as a replacement to the classroom experience.

Participants can expect the following:

- Recordings will be provided the Monday after class is completed
- Recordings will be provided for 14 days
- Recordings will be accessible as “View Only” status and cannot be copied
- Sharing recordings is strictly prohibited

To request recordings, please fill out the form linked in Learn++.

Thank you for your understanding and adhering to the policy.

Note About Virtual Trainings

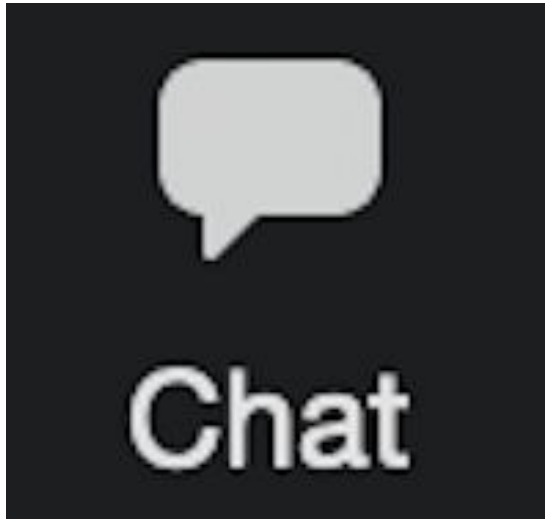
What we want

...what we've got

Virtual Training Expectations for You

Arrive on time / return on time

Mute unless speaking



Use chat or ask questions verbally

Virtual Training Expectations for Me

I pledge to:

- Make this as interesting and interactive as possible
- Ask questions in order to stimulate discussion
- Use whatever resources I have at hand to explain the material
- Try my best to manage verbal responses so that everyone who wants to speak can do so
- Use an on-screen timer for breaks so you know when to be back

Prerequisites

Develop Intelligence

- Solid understanding of the Java programming language to Java 8 including lambda expressions
- Good understanding of basic Stream concepts, including filter, map, flatMap, and reduce

Objectives

Develop Intelligence

At the end of this course you will be able to:

- Write code that conforms to the requirements for reliable execution in a parallel stream
- Write code using the three-argument collect method of a stream
- Describe the requirements for operations in a parallel reduction / collection situation
- Explain the costs of running a stream in ordered mode
- Express the architectural considerations that influence whether a parallel stream mode will be beneficial or not

About you

In 90 seconds!

- What you hope to learn
- What your background level is

THANK YOU

Develop Intelligence

Parallel Stream Operations

Develop Intelligence

If operations on individual data items are entirely independent of one another, these operations can be performed by concurrent threads, provided that the infrastructure handles data-visibility (*happens-before*) requirements.

Java's Streams API supports this directly. Operations on the stream elements should generally be *non-interfering* and *stateless*.

Non-interfering means an operation does not affect the source of data.

Thread safety for parallel streams

Develop Intelligence

Generally, operations should

- Not mutate existing data, but create new data for results
- Avoid visible side-effects
- Use only their own arguments as input

Parallel collections

Java's `collect` operation is a variation of a reduce operation, modified to mutate data, rather than continually create new objects to represent intermediate results. The effect is to reduce object allocation and initialization load on the CPUs and also reduce load on the garbage collector.

For such an approach to be safe, the objects being mutated must either be thread-safe (which is likely to create thread contention in the general case) or thread-confined.

The behavior of Java's three-argument `collect` method is built around thread-confined mutable objects.

A mutating version of reduce

Develop Intelligence

Each thread must have its own work-in-progress mutable result

The system decides how many threads, so we must provide a means of building those result objects, hence a `Supplier<R>` for a result type R.

The second operation merges data one at a time into the current thread's result object.

The third operation merges one thread's result into that of another. This will be used only for parallel streams and will be called enough times to get all the accumulated results into a single result object.

Constraints on reduce/collect

Develop Intelligence

Reduce/collect operations generally should be associative.

If the operation is also commutative, then order need not be maintained in the stream.

Order is generally maintained if the source and collector are ordered, but this can be disabled using `mystream.unordered()`.

Architectural considerations for parallel operation

Develop Intelligence

Concurrency always involves some additional overhead. Throughput gains might depend on ratio of real computation work to that overhead.

If your process completes faster, parallel might be an option, but would you be taking those CPUs away from other equally important work? If so it's unlikely to be a smart choice to run parallel. In a multi-client server system, those CPUs usually do have something else important to do.

Maintaining ordered in a streams can be hugely expensive if many items arrive substantially out of order.

If the arrival rate of items from the streams is low relative to the time taken to update results, a single threadsafe / synchronized result object might be preferred.